Mapping the Internal Dynamics of Large Language Models Using Weighted Tethers

30 December 2024
Simon Edwards
Research Acceleration Initiative

Introduction

The core advantage of the use of pre-trained neural network models, particularly with “Large Language Models” is that the bulk of the computational heft may be performed ahead of time, enabling large number of queries against the model at relatively low computational cost.

This system has as its weakness that deficiencies in the system are not correctable. Once the model is constructed, much like a compiled program, it can’t be modified without decompilation, modification and recompilation. What’s more, the nature of even common deficiencies in LLMs are not understood by the community at large. Perhaps the most well-known of these deficiencies is the problem of ‘hallucination,’ which is a buzzword which refers to the fact that like many humans, the LLM “reaches” and always prefers to provide entirely erroneous answers to no answer whatsoever.

Abstract

In order to prevent problems such as hallucinations, one needs to be able to determine where within the overall logic matrix a fault is occurring and to make an ad-hoc modification to that system which is specific to the query being made; ideally without introducing too great a computational load. It would be impracticable to re-train a pre-trained algorithm for each and every query and, therefore, the goal ought to be to trace the origins of specific, common problems such as the hallucination problem and to formulate standardized fixes which allow for a series of modifications to the core LLM to be applied and for multiple outputs to be generated according to each of a series of modified models.

These modifications may be applied in real-time to compiled LLMs through the injection of instructions directly into RAM at specific times during the overall computational process at which particular pieces of logic can be expected be presently under active handling and through heuristic analysis of the data being handled. In order for data/values/weights to be “washed out,” for example, other values must be duplicated repeatedly within adjacent logic trees in order to justify that action. This sort of decision about which instructions to discard and which to maintain is carried out by humans routinely. For example, if a human is asked to carry out several activities at once, they will eventually not merely fail to keep up with all of the assigned tasks, but they will conveniently forget about one of them until reminded. Within an LLM, it becomes clear than this forgetfulness is actually the consequence of those tasks which are readily comprehended by the neural network taking up increasingly large proportions of the overall computational capacity of the neural network, be it biological or that of an Artificial Intelligence. By identifying duplicated logic sequences within RAM

heuristically, this improper logic-washing can be temporally demarcated. From there, the logic being washed out can be preserved by an overarching system and re-inserted at the top of the logic tree so that it may be properly considered.

In order to create a map of the internal dynamics of such a system, I propose that input terms such as whole words be purposefully duplicated by prime values (not by simply repeating the input words but through logical re-injection of the terms directly into the actively handled data) in order to bring about duplication of output values as well as to create traceable markers which allow for patterns to of logic to be followed through the system via heuristic analysis. Although essentially encrypted, these patterns would remain discernible as a result of their unique length. Although the length of bytes occupied in memory, they may continue to be traced through many computational cycles as their prevalence would tend to be mitigated or amplified by predictable proportions in each cycle which are related to the overall size of the model. We may also observe whence systems of logic interact with one another within the system by looking for values which are factors of two of the distinctly injected values. If we've artificially stepped up the prevalence of one logic cluster by a value of three-fold (prime) and another cluster by five-fold (prime,) and we find that a daughter logic cluster occupies a data volume which is 15-fold the original size, we could then deduce that the triplicated cluster interacted with the quintupled cluster.

Sometimes, inputs are entirely ignored. For example, if I ask an LLM for a list of journalists specializing in stories concerning technology, it will generally return a list of the most prominent web search results. I can ask the model to return additional names and the model will return a mixture of different names including many which have been listed previously. The model will never return an exhaustive list because it is weighting the results according to their prominence in a web search and the number of times the name appears in the web search. If I tell the LLM explicitly not to list any names which were listed in previous responses, the LLM will disregard this imperative and rather than giving a response such as, "There are no more because those were all the names I could find," the LLM will repeatedly provide the same list of names rather than the more esoteric results I am seeking. This willful ignorance is the other side of the 'hallucination' coin. LLMs are capable of ignoring specific instructions because in order for the model to function at all, conflicts must be remedied by pruning outlying weights with values which far exceed computational norms for a single node. Sometimes, this pruning is executed inappropriately, as in the case when instructions such as the aforementioned instruction are disregarded.

If we could identify the specific logic nodes which are being either improperly pruned by the overall logic within the so-called "black box," adjustments could be made in order to protect the logical sovereignty of that node for one additional cycle or for as many as several additional cycles. In cases in which a hallucination is occurring, the problem may be remedied by de-emphasizing the need to produce an output regardless of whether the needed data exists within the system or not. With these adjustments, the LLM can learn to concede the limitations of its own knowledge. While this may be beyond the capacity of some humans, for humans, this is a shortcoming of character rather than of logic.

By probing the internal workings of the model through the duplication of inputs in a manner which allows for interactions of different values to be traced throughout the computational process, we may begin to experiment with in-situ modification of LLMs in order to correct for errors on-the-fly.

Conclusion

LLMs may, conceivably, be modified using the aforementioned approach in order to entirely eliminate these common faults and to eliminate other, less common faults by providing a variety of alternative builds which may be tried by the end user. Although efficiency is a core strength of LLMs, by introducing logical inefficiency at strategic points in the computational process, these purposefully introduced inefficiencies may be used much as radiological tracers are used in medical imaging in order to identify areas of dysfunction. Common problems may be corrected and the models may be revised based upon what is learned through this sort of probing of the models.